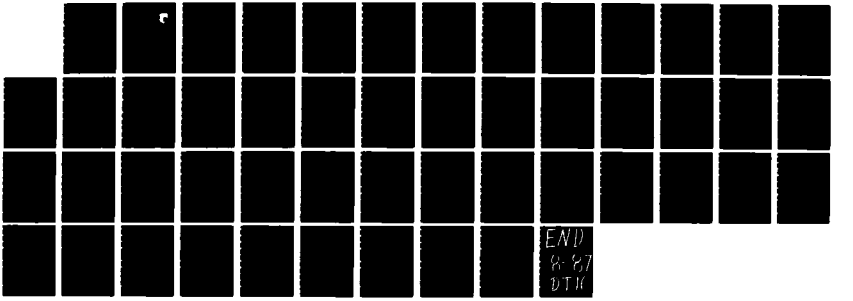
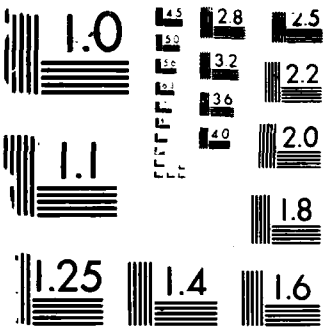
AD-A182 588 INTEGRATED INFORMATION SUPPORT SYSTEM (IISS) VOLUME 8 1/1
USER INTERFACE SUBS (U) GENERAL ELECTRIC CO
SCHENECTADY NY PRODUCTION RESOURCES CONSU
UNCLASSIFIED M BUTTERWORTH ET AL 01 NOV 85 DS-620144501 F/G 12/5 NL





AFWAL-TR-86-4006
Volume VIII
Part 20

INTEGRATED INFORMATION SUPPORT SYSTEM (IISS)
Volume VIII - User Interface Subsystem
Part 20 - Report Writer Development Specification

General Electric Company
Production Resources Consulting
One River Road
Schenectady, New York 12345

Final Report for Period 22 September 1980 - 31 July 1985

November 1985

MATERIALS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AFB, OH 45433-6533

NOTICE

When Government drawings specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever, and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto

This report has been reviewed by the Office of Public Affairs (ASD/PA) and is releasable to the National Technical Information Service (NTIS) At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

DAVID L JUDSON, PROJECT MANAGER
AFWAL/MLTC
WRIGHT PATTERSON AFB OH 45433

5 Aug 1986
DATE

FOR THE COMMANDER

GERALD C. SHUMAKER, BRANCH CHIEF
AFWAL/MLTC
WRIGHT PATTERSON AFB OH 45433

7 Aug 86
DATE

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFWAL/MLTC, W-PAFB, OH 45433 to help us maintain a current mailing list."

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION: Unclassified	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S): AFWAL-TR-86-4006 Vol VIII, Part 20
6a NAME OF PERFORMING ORGANIZATION: General Electric Company Production Resources Consulting; 6b OFFICE SYMBOL (If applicable)	7a NAME OF MONITORING ORGANIZATION: AFWAL/MLTC
6c ADDRESS (City, State and ZIP Code): 1 River Road Schenectady, NY 12345	7b ADDRESS (City, State and ZIP Code): WPAFB, OH 45433-6533
8a NAME OF FUNDING/SPONSORING ORGANIZATION: Materials Laboratory Air Force Systems Command, USAF; 8b OFFICE SYMBOL (If applicable): AFWAL/MLTC	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: F33615-80-C-5155
8c ADDRESS (City, State and ZIP Code): Wright-Patterson AFB, Ohio 45433	10 SOURCE OF FUNDING NOS: PROGRAM ELEMENT NO. 78011F; PROJECT NO. 7500; TASK NO 62; WORK UNIT NO 01
11 TITLE (Include Security Classification): (See Reverse)	
12 PERSONAL AUTHOR(S): Butterworth, Manning and Glandorf, Frank	
13a TYPE OF REPORT: Final Technical Report; 13b TIME COVERED: 22 Sept 1980 - 31 July 1985; 14 DATE OF REPORT (Yr., Mo., Day): 1985 November; 15 PAGE COUNT: 48	
16 SUPPLEMENTARY NOTATION: ICAM Project Priority 6201	The computer software contained herein are theoretical and or references that in no way reflect Air Force-owned or -developed computer software.

17 COSATI CODES		
FIELD	GROUP	SUB GR
1308	0905	

18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

This DS establishes the requirements for a computer program identified as the Report Writer that translates report definitions into programs that access data bases via the CDM and report the extracted data in a formatted way usually with interspersed identifying text and possibly statistical summaries. The destination of the reports is some hardcopy medium such as lineprinter output.

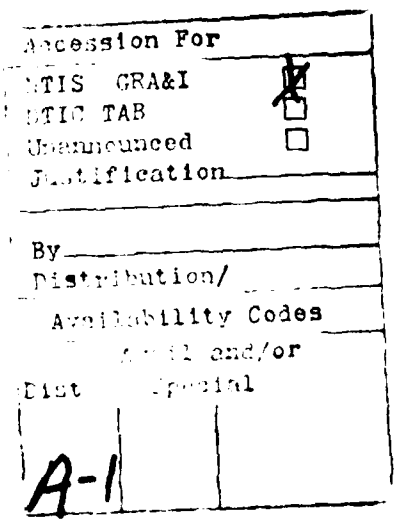
20 DISTRIBUTION AVAILABILITY OF ABSTRACT: UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS ☐	21 ABSTRACT SECURITY CLASSIFICATION: Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL: David L Judson	22b TELEPHONE NUMBER (Include Area Code): 513-255-6976; 22c OFFICE SYMBOL: AFWAL/MLTC

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE

11. Title

Integrated Information Support System (IISS)
Vol VIII - User Interface Subsystem
Part 20 - Report Writer Development Specification

A S D 86 0036
9 Jan 1986



PREFACE

This development specification covers the work performed under Air Force Contract F33615-80-C-5155 (ICAM Project 6201). This contract is sponsored by the Materials Laboratory, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio. It was administered under the technical direction of Mr. Gerald C. Shumaker, ICAM Program Manager, Manufacturing Technology Division, through Project Manager, Mr. David Judson. The Prime Contractor was Production Resources Consulting of the General Electric Company, Schenectady, New York, under the direction of Mr. Alan Rubenstein. The General Electric Project Manager was Mr. Myron Hurlbut of Industrial Automation Systems Department, Albany, New York.

Certain work aimed at improving Test Bed Technology has been performed by other contracts with Project 6201 performing integrating functions. This work consisted of enhancements to Test Bed software and establishment and operation of Test Bed hardware and communications for developers and other users. Documentation relating to the Test Bed from all of these contractors and projects have been integrated under Project 6201 for publication and treatment as an integrated set of documents. The particular contributors to each document are noted on the Report Documentation Page (DD1473). A listing and description of the entire project documentation system and how they are related is contained in document FTR620100001, Project Overview.

The subcontractors and their contributing activities were as follows:

TASK 4.2

Subcontractors	Role
Boeing Military Aircraft Company (BMAC)	Reviewer.
D. Appleton Company (DACOM)	Responsible for IDEF support, state-of-the-art literature search.
General Dynamics/ Ft. Worth	Responsible for factory view function and information models.

Subcontractors	Role
Illinois Institute of Technology	Responsible for factory view function research (IITRI) and information models of small and medium-size business.
North American Rockwell	Reviewer.
Northrop Corporation	Responsible for factory view function and information models.
Pritsker and Associates	Responsible for IDEF2 support.
SofTech	Responsible for IDEF0 support.

TASKS 4.3 - 4.9 (TEST BED)

Subcontractors	Role
Boeing Military Aircraft Company (BMAC)	Responsible for consultation on applications of the technology and on IBM computer technology.
Computer Technology Associates (CTA)	Assisted in the areas of communications systems, system design and integration methodology, and design of the Network Transaction Manager.
Control Data Corporation (CDC)	Responsible for the Common Data Model (CDM) implementation and part of the CDM design (shared with DACOM).
D. Appleton Company (DACOM)	Responsible for the overall CDM Subsystem design integration and test plan, as well as part of the design of the CDM (shared with CDC). DACOM also developed the Integration Methodology and did the schema mappings for the Application Subsystems.

Subcontractors	Role
Digital Equipment Corporation (DEC)	Consulting and support of the performance testing and on DEC software and computer systems operation.
McDonnell Douglas Automation Company (McAuto)	Responsible for the support and enhancements to the Network Transaction Manager Subsystem during 1984/1985 period.
On-Line Software International (OSI)	Responsible for programming the Communications Subsystem on the IBM and for consulting on the IBM.
Rath and Strong Systems Products (RSSP) (In 1985 became McCormack & Dodge)	Responsible for assistance in the implementation and use of the MRP II package (PIOS) that they supplied.
SofTech, Inc.	Responsible for the design and implementation of the Network Transaction Manager (NTM) in 1981/1984 period.
Software Performance Engineering (SPE)	Responsible for directing the work on performance evaluation and analysis.
Structural Dynamics Research Corporation (SDRC)	Responsible for the User Interface and Virtual Terminal Interface Subsystems.

Other prime contractors under other projects who have contributed to Test Bed Technology, their contributing activities and responsible projects are as follows:

Contractors	ICAM Project	Contributing Activities
Boeing Military Aircraft Company (BMAC)	1701, 2201, 2202	Enhancements for IBM node use. Technology Transfer to Integrated Sheet Metal Center (ISMC).

Contractors	ICAM Project	Contributing Activities
Control Data Corporation (CDC)	1502, 1701	IISS enhancements to Common Data Model Processor (CDMP).
D. Appleton Company (DACOM)	1502	IISS enhancements to Integration Methodology.
General Electric	1502	Operation of the Test Bed and communications equipment.
Hughes Aircraft Company (HAC)	1701	Test Bed enhancements.
Structural Dynamics Research Corporation (SDRC)	1502, 1701, 1703	IISS enhancements to User Interface/Virtual Terminal Interface (UI/VTI).
Systran	1502	Test Bed enhancements. Operation of Test Bed.

TABLE OF CONTENTS

APPENDICES

FIGURES

SECTION 1

SCOPE

1.1 Identification

This specification establishes the performance, development, test and qualification requirements of a computer program identified as the Report Writer (RW). The RW is one configuration item of the Integrated Information Support System (IISS) User Interface (UI).

1.2 Functional Summary

This Computer Program Configuration Item (CPCI) is used to report selected information stored in the database accessible through the Common Data Model (CDM).

The major functions of the Report Writer are:

1. The placement and formatting of fixed textual information and database information, i.e. CDM data.
2. The summarization of simple statistical attributes of the reported information such as counts, sums, and averages.
3. The retrieval of data from the CDM.

SECTION 2

DOCUMENTS

2.1 Reference Documents

[1] Structural Dynamics Research Corporation, Terminal Operator Guide, OM 620144000 , 1 November 1985.

[2] Structural Dynamics Research Corporation, IISS Form Processor User Manual, UM 620144200B, 1 November 1985.

[3] General Electric Company, System Design Specification, 7 February 1983.

[4] Structural Dynamics Research Corporation, Forms Language Compiler Development Specification, DS 620144401B, 1 November 1985.

[5] Structural Dynamics Research Corporation, Forms Driven Form Editor Development Specification, DS 620144402B, 1 November 1985.

[6] Structural Dynamics Research Corporation, Report Writer Development Specification, DS 620144501 , 1 November 1985.

[7] Structural Dynamics Research Corporation, Rapid Application Generator Development Specification, DS 620144502 , 1 November 1985.

[8] Structural Dynamics Research Corporation, Text Editor Development Specification, DS 620144600B, 1 November 1985.

[9] Structural Dynamics Research Corporation, Application Interface Development Specification, DS 620144700 , 1 November 1985.

[10] Structural Dynamics Research Corporation, User Interface Services Development Specification, DS 620144100B, 1 November 1985.

[11] Structural Dynamics Research Corporation, Form Processor Development Specification, DS 620144200B, 1 November 1985.

[12] Structural Dynamics Research Corporation, Virtual Terminal Interface Development Specification, DS 620144300B, 1 November 1985.

[13] Systran, ICAM Documentation Standards, 15 September 1983.

[14] Systran, User's Manual for the ICAM Integrated Support System (IISS) Neutral Data Manipulation Language (NDML), February, 1983.

[15] Systran, Implementation of Enhancements of NDML SELECT COMMAND, 25 July 1984, revised 9 September 1984.

[16] Systran, Discussion of Function Implementation NDML SELECT COMMAND, 25 July 1984, revised 4 September 1984.

2.2 Terms and Abbreviations

Application Generator: (AG), subset of the IISS User Interface that consists of software modules that generate IISS application code and associated form definitions based on a language input. The part of the AG that generates report programs is called the Report Writer. The part of the AG that generates interactive applications is called the Rapid Application Generator.

Application Interface: (AI), subset of the IISS User Interface that consists of the callable routines that are linked with applications that use the Form Processor or Virtual Terminal. The AI enables applications to be hosted on computers other than the host of the User Interface.

Application Process: (AP), a cohesive unit of software that can be initiated as a unit to perform some function or functions.

Attribute: field characteristic such as blinking, highlighted, black, etc. and various other combinations. Background attributes are defined for forms or windows only. Foreground attributes are defined for items. Attributes may be permanent, i.e., they remain the same unless changed by the application program, or they may be temporary, i.e., they remain in effect until the window is redisplayed.

Common Data Model: (CDM), IISS subsystem that describes com data application process formats, form definitions, etc. of the IISS and includes conceptual schema, external schemas, internal schemas, and schema transformation operators.

Computer Program Configuration Item: (CPCI), an aggregation of computer programs or any of their discrete portions, which satisfies an end-use function.

Conceptual Schema: (CS), the standard definition used for all data in the CDM. It is based on IDEF1 information modelling.

Device Drivers: (DD), software modules written to handle I/O for a specific kind of terminal. The modules map terminal specific commands and data to a neutral format. Device Drivers are part of the UI Virtual Terminal.

Display List: is similar to the open list, except that it contains only those forms that have been added to the screen and are currently displayed on the screen.

External Schema: (ES), an application's view of the CDM's conceptual schema.

Field: two-dimensional space on a terminal screen.

Form: structured view which may be imposed on windows or other forms. A form is composed of fields. These fields may be defined as forms, items, and windows.

Form Definition: (FD), forms definition language after compilation. It is read at runtime by the Form Processor.

Forms Definition Language: (FDL), the language in which electronic forms are defined.

Forms Driven Form Editor: (FDFE), subset of the FE which consists of a forms driven application used to create Form Definition files interactively.

Form Editor: (FE), subset of the IISS User Interface that is used to create definitions of forms. The FE consists of the Forms Driven Form Editor and the Forms Language Compiler.

Form Hierarchy: a graphic representation of the way in which forms, items and windows are related to their parent form.

Forms Language Compiler: (FLAN), subset of the FE that consists of a batch process that accepts a series of forms definition language statements and produces form definition files as output.

Form Processor: (FP), subset of the IISS User Interface that consists of a set of callable execution time routines available to an application program for form processing.

IISS Function Screen: the first screen that is displayed after logon. It allows the user to specify the function he wants to access and the device type and device name on which he is working.

Integrated Information Support System: (IISS), a test computing environment used to investigate, demonstrate and test the concepts of information management and information integration in the context of Aerospace Manufacturing. The IISS addresses the problems of integration of data resident on heterogeneous data bases supported by heterogeneous computers interconnected via a Local Area Network.

Item: non-decomposable area of a form in which hard-coded descriptive text may be placed and the only defined areas where user data may be input/output.

Message: descriptive text which may be returned in the standard message line on the terminal screen. They are used to warn of errors or provide other user information.

Message Line: a line on the terminal screen that is used to display messages.

Network Transaction Manager: (NTM), IISS subsystem that performs the coordination, communication and housekeeping functions required to integrate the Application Processes and System Services resident on the various hosts into a cohesive system.

Neutral Data Manipulation Language: (NDML), the command language by which the CDM is accessed for the purpose of extracting, deleting, adding, or modifying data.

Operating System: (OS), software supplied with a computer which allows it to supervise its own operations and manage access to hardware facilities such as memory and peripherals.

Page: instance of forms in windows that are created whenever a form is added to a window.

Paging and Scrolling: a method which allows a form to contain more data than can be displayed with provisions for viewing any portion of the data buffer.

Physical Device: a hardware terminal.

Presentation Schema: (PS), may be equivalent to a form. It is the view presented to the user of the application.

Qualified Name: the name of a form, item or window preceded by the hierarchy path so that it is uniquely identified.

Subform: a form that is used within another form.

User Interface: (UI), IISS subsystem that controls the user's terminal and interfaces with the rest of the system. The UI consists of two major subsystems: the User Interface Development System (UIDS) and the User Interface Management System (UIMS).

User Interface Development System: (UIDS), collection of IISS User Interface subsystems that are used by applications programmers as they develop IISS applications. The UIDS includes the Form Editor and the Application Generator.

Window: dynamic area of a terminal screen on which predefined forms may be placed at run time.

SECTION 3

REQUIREMENTS

3.1 Computer Program Definition

The Report Writer is used to translate report definitions into programs that access data bases via the CDM and report the extracted data in a formatted way usually with interspersed identifying text and possibly statistical summaries. The destination of the reports is some hardcopy medium such as lineprinter output.

The Report Definition Language includes the Form Definition Language and the CDM Neutral Data Manipulation Language (NDML). The FDL is used to create the report layout and the NDML is used to map the CDM data to the report forms.

3.1.1 System Capacities

The RW is written in C and COBOL and runs on a DEC VAX minicomputer under the VMS operating system.

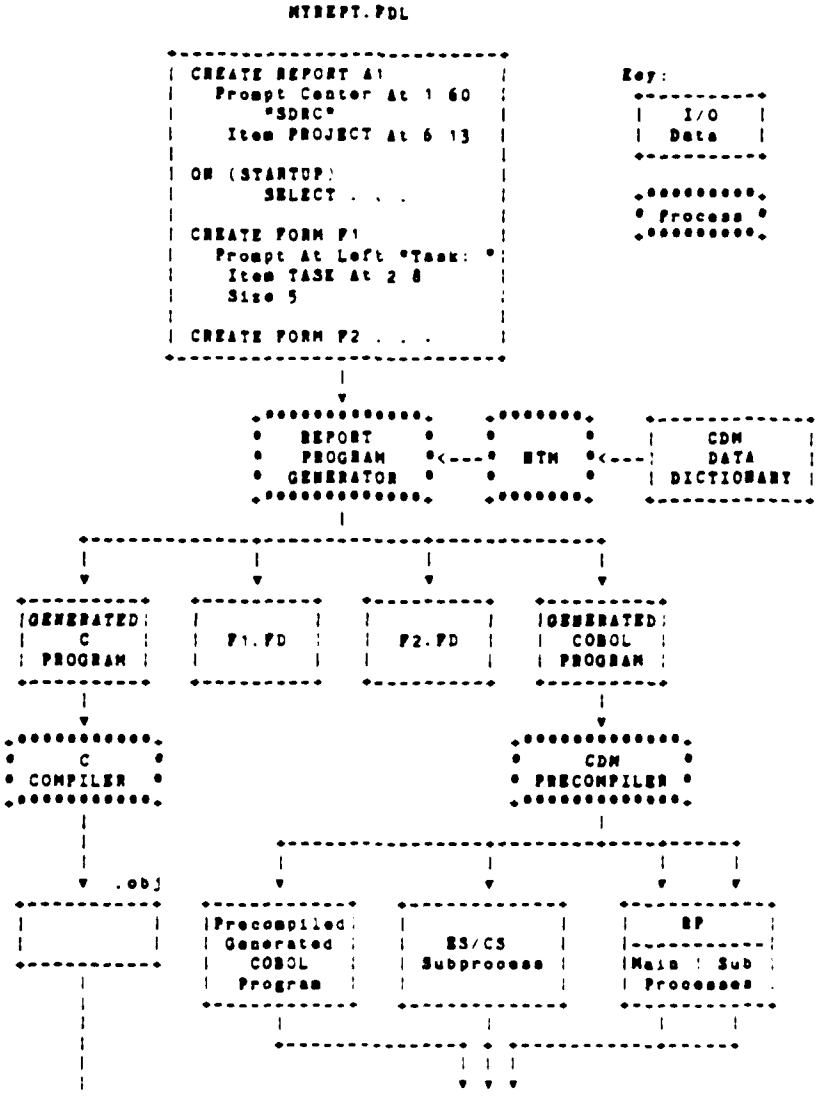
3.1.2 Interface Requirements

The COBOL program output by the RW is constrained to be compatible with statement forms expected by the CDM precompiler.

The syntax of the Report Definition Language accepted as input to FLAN is modeled after the Forms Definition Language and the Neutral Data Manipulation Language. The Forms Definition Language is used to describe the report layout but additional capabilities such as pagebreaking, overflow conditions as needed for controlling report output have been added to the language syntax. Also, the language syntax for mapping the CDM data to the report form items is accomplished by NDML.

3.1.2.1 Interface Block Diagram

The interface block diagram for the Report Writer is shown in Figure 3-1.



Continued on next page

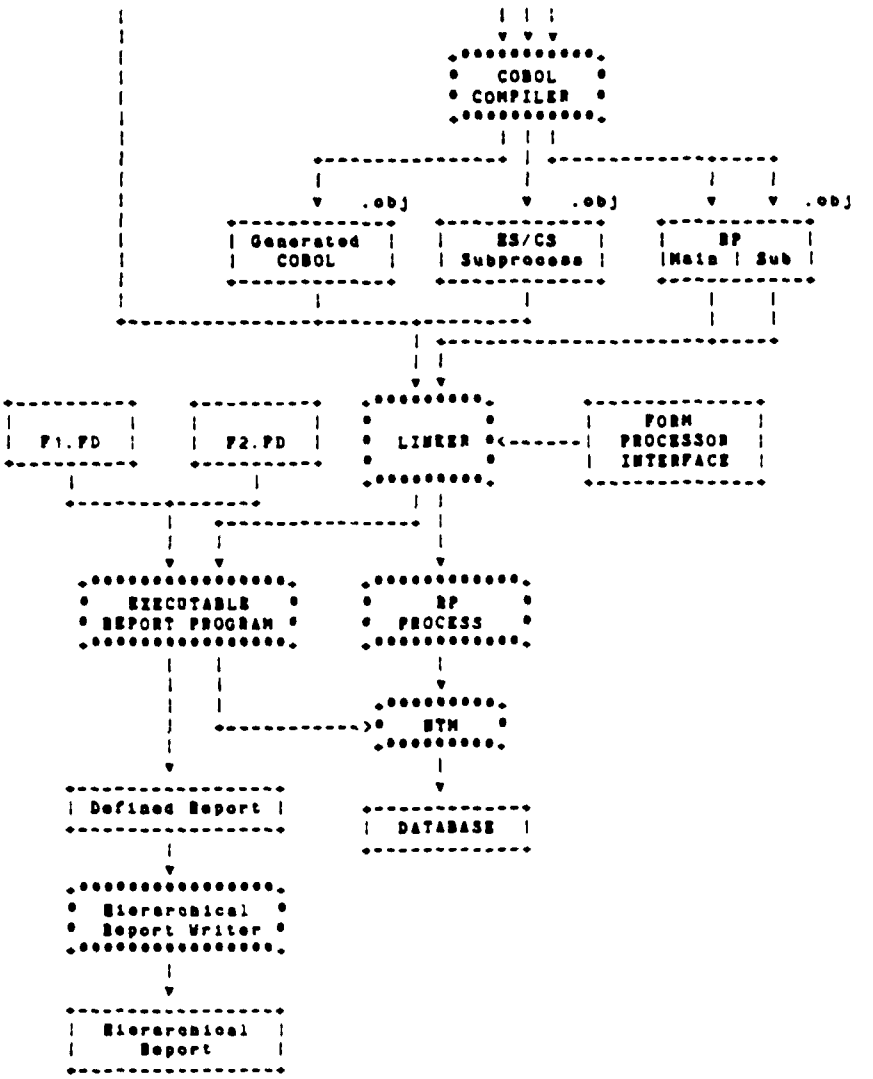


Figure 3-1. Report Writer Interface Block Diagram

3.1.2.2 Detailed Interface Definition

The syntax of the Report Definition Language (RDL), is documented in Appendix C. This language is intended to provide access to all Report Writer functionality.

The collection of RDL entries that define a report application is a report application definition. A report application definition is written to an RDL source file with any text editor one might use to prepare a program source file.

The RDL source file is processed by the Report Writer Generator to produce separate binary Form Definition (FD) files for each form definition and to generate C and COBOL code which together with the Form Processor (FP) procedures and data dictionary information contained in the Common Data Model (CDM) provide the final report application.

Since the COBOL program contains NDML statements to acquire the data via the CDM, it must be preprocessed by the CDM precompiler. After compilation and linking the program is executed to produce the report.

If the report is a Hierarchical Report, it is further processed by the Hierarchical Report Writer (HRW) which rearranges it into the appropriate tree structure.

3.2 Detailed Function Requirements

The RW exploits the versatility of the Forms Definition Language by extending it in needed ways. The reports produced by the RW are similar to forms but differ in two ways. First, they are not interactive when executed; and, second, they are not scrolled or paged in the same way.

The RW generator produces the report application from the database by applying the instantiation rules described in Appendix A to the data structures created by the action of FLAN upon the RDL source file. The report application produces the report by accessing the CDM and outputting the data using form processor routines

When a processing error occurs, an error message is output. The error message includes as much information as possible about the nature of the error and the location of the report application statement causing it. If at all possible, processing continues after an error is recognized so that additional errors may also be detected.

3.2.1 User's View of the Report

A user should think in terms of what he would like a page to look like and how the page repeats. The use of conditions and actions can be used to shape what the report actually will look like with data.

3.2.2 Textual Output

Text may be associated with the report (form), sub-forms, or field items. The defined text is unchanging throughout the report. Its placement is governed by giving a location as in the FDL.

3.2.3 Stretchy Lines

The developer using this CPCI shall be able to define extensible lines associated with forms and items. The starting and ending locations mean the centers of the given character positions. By using relative positions for the ending locations of lines, the developer can define lines which effectively stretch to correspond to the length or width of the associated field as it expands at run time.

The exact representation of the lines is device dependent. On a particular device the mechanism with the highest resolution will be the one implemented. If the device only supports typewriter graphics, then the dash will be used to construct horizontal lines and the vertical bar (if available) or the exclamation point will be used to construct vertical lines. At the juncture or intersection of two lines a plus sign will replace the vertical and horizontal characters.

3.2.4 Statistical Summarization

Certain simple statistical computations may be performed upon item values and included in the output. These statistical summaries may be included at the point where an exceptional condition arises such as the change of an item field value or anywhere else. Each item field definition may specify a count.

sum, average, minimum, and maximum to be computed and assigned to another item field. The calculated results are always available and may be output at any time. When the results are reported, the calculations are reset.

3.2.5 Picture Specifications

Each item field may have a picture specified to define the output format, for example, to define the position of a decimal point or to include a dollar sign before a decimal number. More specifically the editing permitted includes numeric and alphanumeric specification, leading zero suppression, decimal placement, leading sign indicators, currency symbols, and placement of embedded commas. The picture specification applies to items that are mapped to CDM data only.

3.2.6 Nonduplication of Item Values

The output of unchanged item values may be suppressed by including the NODUP option with the ITEM statement defining the data field.

3.2.7 Exceptional Conditions

Tests can be established for certain exceptional conditions, viz. when the addition of an array element would cause the array to exceed the bounds of a containing form or when an item field changes in value. An action or group of actions can be specified to occur when the condition arises. The action may be to signal another condition, to repeat the appearance of a form, or to assign a value to an item, for example.

Practical applications of this facility include output of headers, footers, statistical summaries and paging.

The semantics of conditions and actions are given in Appendix B.

3.2.7.1 CHANGE Condition

The change of an item field value can be treated as an exceptional condition which may cause one or more actions to occur. A change of value means a change of value when the items are examined in the completed report in the order in which they are defined to appear by the report definition.

To be meaningful this facility implies that the item values have been sorted before inclusion in the report (probably by use of the ORDER option of the SELECT statement used to extract the data from the CDM).

3.2.7.2 OVERFLOW Condition

This condition is occurs when attempting to add an element to an open ended array that would exceed the bounds of a containing form. An open ended array is one which has a star (e.g. '*') specified as the number of elements. The name of the array element that would cause the overflow is part of the condition.

3.2.7.3 STARTUP Condition

A set of actions which are performed when the report is started.

3.2.7.4 Exception Actions

3.2.7.4.1 PAGE Option

When the condition to which this action is attached occurs, this option causes a physical page eject.

3.2.7.4.2 PRESENT Option

This option begins the instantiation of the specified form (instantiation rules are listed in appendix A). Data which was obtained through the prior use of a SELECT option is moved to the forms.

PRESENT has two syntactical forms. In the first a form name and possibly a window are specified. This causes the removal of any form in the specified window and adds the specified form to the specified window and then instantiation of the form. If no window is specified then 'screen;' is used. In the second syntactical form, a qualified name of a form is given. This simply begins instantiation of the form which is already in the form hierarchy.

3.2.7.4.3 SET Option

This option allows the setting of an item field to a specified value, either an integer or string constant.

3.2.7.4.4 SIGNAL OVERFLOW Option

This option allows one exceptional condition to be used to trigger another exceptional condition.

3.2.7.4.5 SELECT Option

This option describes what information is desired from the CDM and how it maps to ITEMs on the forms. SELECT will perform a query and make the results availble. A PRESENT action must then be performed to move the data into the forms.

3.2.8 Instantiation Rules

Instantiation of fields on the report is controlled by a set of rules given in Appendix A.

3.2.9 Graphical Display

The Report Writer will provide the capability to graphically display data in two-dimensional format using common business graphics. The RW will provide this through a field, the graph field, which may contain a single graph of predefined type and structure. The available types are bar chart, pie chart, and x-y plot. In the case of bar charts and x-y plots more than one data set may be plotted in a single graph yielding multiple curves or parallel or stacked bars, for example.

Size and location values are in terms of the default terminal character sizes and positions. No scaling is done automatically; it is the user's responsibility to ensure that textual information, for example, is of the proper size to fit in the available space. Portions of a graph which do not fit in a displayed space are clipped, not wrapped. The RW will attempt to adjust graphs appropriately for the aspect ratio of the device on which they are displayed so that, for example, pie charts are circles regardless of the display device.

Optional attributes will have reasonable default values so that a graph can be included in a display with a minimum of effort at specification. Attributes unsupported on a given device, for example color on a monochrome device, will default to appropriate values.

On devices supporting them, capabilities shall include the choice of line style and width (e.g. solid, dashed, dotted, thin, and thick), and the choice of color of the curve, the background, and the area under the curve. Other choices shall always be available including the specification of the size and font style of text in axis labels, tick mark labels, and legend entries, the lower and upper limits of axes, the scale (linear or logarithmic) of the axes, the presence or absence of a grid, the sizes of the margins between the borders of the graph field and the graph, some shading pattern below the curves, symbols to appear at the data points, and for pie charts the amount of explosion of individual segments as well as the appearance and location (inside or outside) of the actual numerical values and/or the percentages represented by each segment. For graphs consisting of more than one curve or more than one set of bars there shall also be the option of displaying the data additively or absolutely as measured on the vertical axis.

Graphs shall be for output display only. In particular, although the cursor position may be detected within the graph field, no smaller portion of the graph field may be picked, nor may any data points be picked or altered interactively.

The data sources for the graphs denoted by ordinate and/or abscissa in the language syntax are numeric repeating item fields. The item fields may or may not be displayed as form fields. If they are input fields, then data in them can be altered and the next time the graph using those fields is displayed it will reflect the new values. Graph fields cannot be be presented in windows because the information defining them appears in part in the field definition of the CREATE FORM statement and in part in the CREATE GRAPH statement.

The following paragraphs explain the language features describing graphs in somewhat more detail.

3.2.9.1 Graph Definition

The graph field is defined as a field on a form in the same way as other fields. The graph field must also appear in a CREATE GRAPH statement. The size and location are required parts of the graph field definition. All other attributes of the graph field definition are optional.

3.2.9.2 Additive Versus Absolute Display

When data are displayed using bar charts or x-y plots, more than one dependent data set may be plotted in a given graph. Two or more curves are distinguishable by color or linestyle for example. The ordinate values can be measured on the vertical axis either relative to the horizontal axis or relative to a curve already displayed. The former method is absolute display and the latter additive display. If the graph is a bar chart, the bars will be displayed side by side if the display is absolute and displayed stacked vertically if the display is additive. By default displays will be absolute.

3.2.9.3 Axis Information

Axis labels and tick mark labels can be specified independently for the horizontal and vertical axes. The text in these labels can be defined to have an available font, color, and size (in units of the terminal's standard character height and width). By default text will have a size of 1 by 1 and a simple stick figure font; the color will be the default contrasting color to the background, e.g. white on black. The color of the axis itself including tick marks can also be specified and may differ from the color of the labels.

The tick mark clause specifies the number of major tick marks and optionally the number of minor divisions between each pair of major tick marks. The tick mark label string comprises a sequence of substrings each delimited by a chosen character not appearing elsewhere in the string. If the number of substrings is fewer than the number of major tick marks, then the tick mark label string is reused from the beginning.

If no label string is defined for an axis, the associated item field name is used as the label. In the case that more than one item field is associated with a vertical axis, the name of the first such item field is used. If no tick mark label string is defined, then the axis is divided into a number of intervals which yields nice tick mark label values. Nice values are defined to be multiples by powers of ten of values in the set { .1, .2, .25, and .5}.

The lower and upper limits and the scale (linear or logarithmic) of the axes can be specified also. If the range of the data exceeds the limits specified, then data outside the range and not displayed. Unspecified limits are set by default to give the smallest range which will span the range of the data

such that the tick mark labels can either be those given by the user or can be ones in the set of nice values as defined in the paragraph above. If the scale is not defined to be logarithmic, then by default it is linear.

3.2.9.4 Background Color

The background of the graph field can be defined to have an available color.

3.2.9.5 Curve Information

For x-y plots and in most instances for bar charts several attributes of the curves or bars may be specified including the color of the curve or bars as well as the color or shading pattern of the area below the curve or bars. On devices supporting the feature the linetype (solid, dashed, etc.) and linewidth may also be specified. For x-y plots symbols may optionally be placed at the data point locations either in addition to or instead of the curve. By default data points are connected by solid line segments in the order of occurrence of the source item field values and no data point symbols appear. The symbols may be chosen from a catalog of predefined symbols and will include minimally the dot, plus sign, asterisk, cross or x, and circle. By default no shading occurs below curves or bars and the color is the background color.

3.2.9.6 Grids

Bar charts and x-y plots can be defined to have grids superimposed upon them. By default no grid is displayed. A grid will connect major tick marks in both directions. If minor tick marks have been specified, then a fine grid will connect minor tick marks. If minor tick marks have been defined for only one axis, then fine grid lines will appear perpendicular to only that axis.

3.2.9.7 Legends

A graph can be defined to have a legend in a specified location. By default no legend will appear. The legend can also optionally be circumscribed by a box. The location value is the location of the upper left hand corner of the legend relative to the graph field. The legend entry is defined in the AXIS or PIE clauses of the CREATE GRAPH statement which may also contain specifications of font, size, and color attributes for the legend entry text.

3.2.9.8 Margins

The amount of space surrounding the graph which is delimited by the axes or by the pie is adjustable. This space can be set by the MARGIN keyword in the GRAPH field definition. The number given is the percentage of the corresponding dimension of the graph field. By default all margins are 10 percent.

3.2.9.9 Pie Charts

There are several language features which apply only to pie charts. Some pie chart attributes are associated with individual segments. The segments are numbered sequentially from 1 beginning at a horizontal radius vector to the right.

The explosion factor which is applied on a segment by segment basis is the percentage of the radius of the pie by which a segment is projected radially outward. By default no segments are exploded.

The data source for a pie chart is a single item field. The segments correspond in order with the elements of the item field. The item field values and/or their corresponding percentages of the whole may be displayed for each segment either inside or outside the segment. The location is the same for all segments, but quantities and percentages need not both be inside or outside. By default neither the quantities nor the percentages will appear.

As for other text, color, size, and font may be defined for quantities, percentages, segment labels, and legend entries.

Finally, each segment may be individually colored or shaded with a pattern. By default no pattern will appear within the segments and the interior color will be the background color.

3.2.9.10 Text

Text may be included anywhere within the graph field for titles, annotation, etc. via the PROMPT clause which is similar to the PROMPT clause generally available within FDL except that here the text attributes include font and size and, if supported by the device, color.

3.2.10 Hierarchical Report Writer

The Hierarchical Report Writer is a post-processor which accepts a report containing graphic representations of elements and connection information as input and rearranges the graphic representations to correspond to the connection information. After rearranging, the resulting tree is either divided into pages (by recursively finding the largest subtree which will fit on a page and moving it onto a separate page) or strips of a specified width and written out.

3.2.11 Data Structures

Figures 3-2a and 3-2b below show the data structures particular to the report writer application generator. The field type data structure is abbreviated in that many structure members have been omitted for clarity. These data structures are created by the Report Writer Generator.

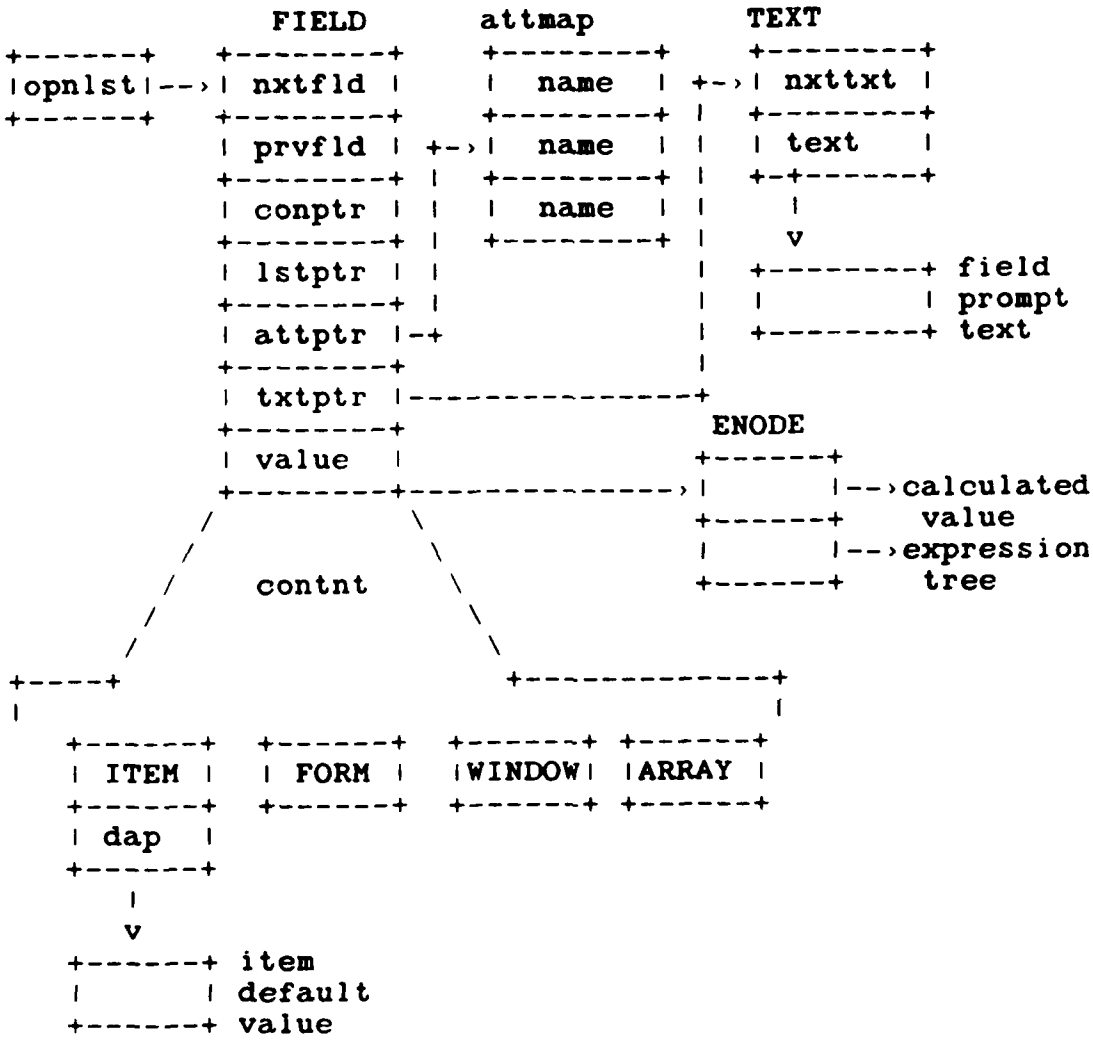


Figure 3-2a Field Data Structures

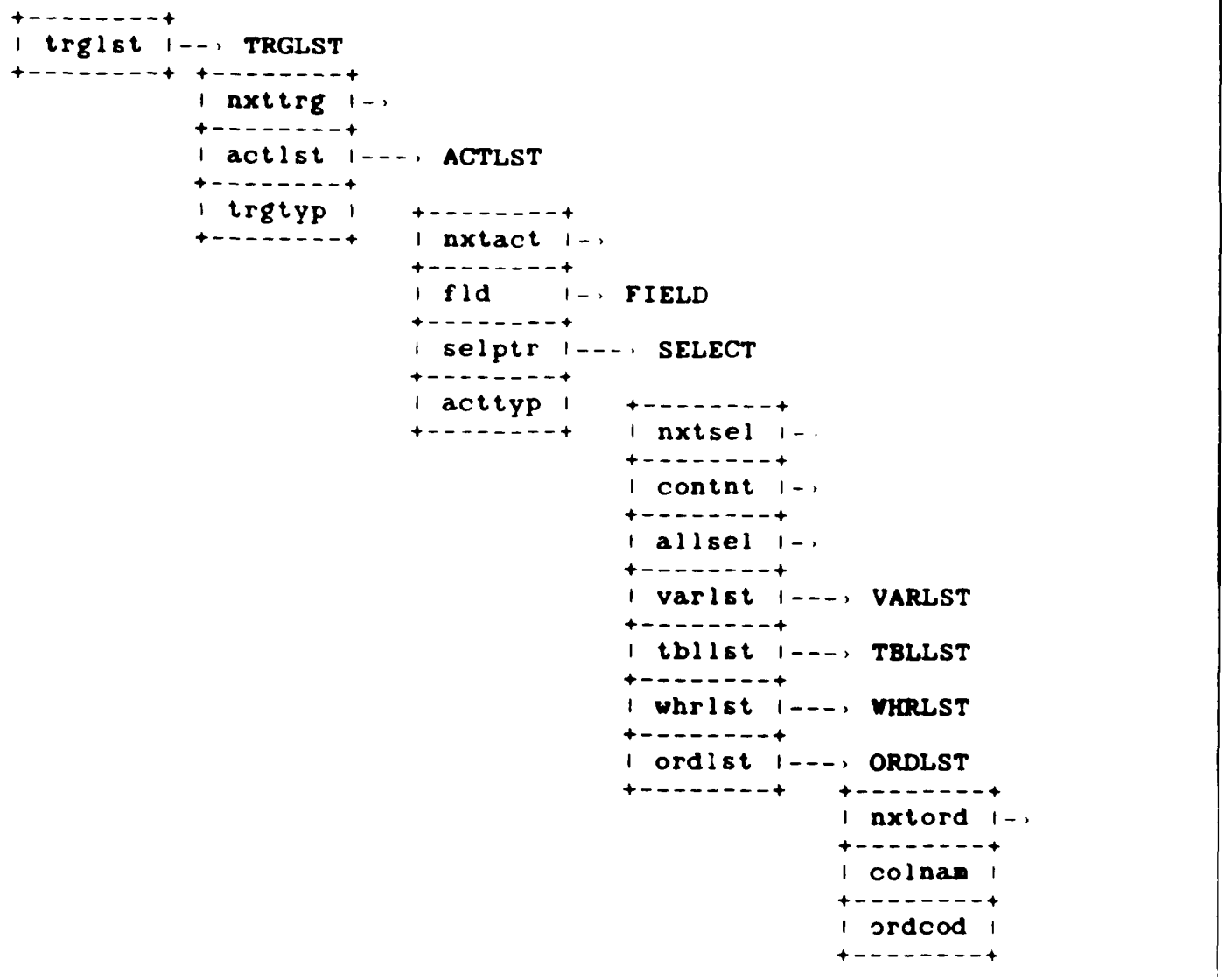


Figure continued on next page.

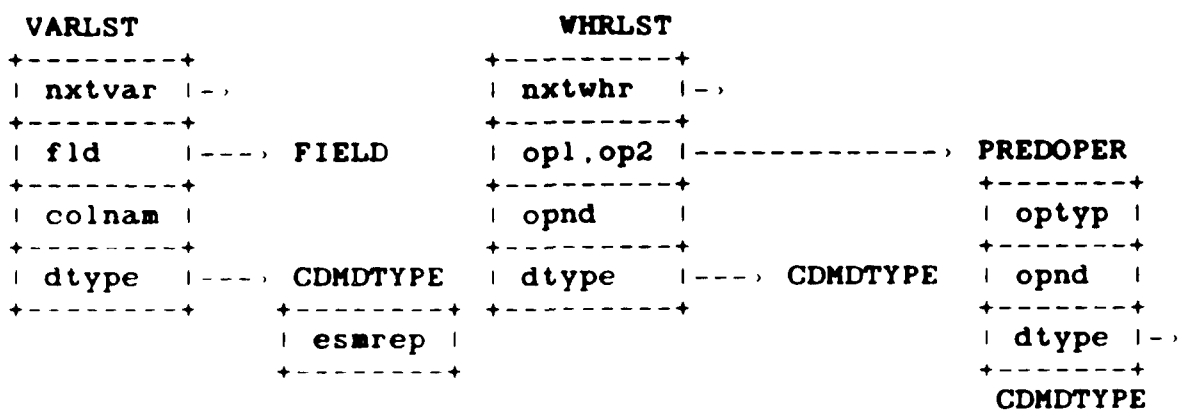


Figure 3-2b Report Generation Data Structures

Figure 3-3 below illustrates the Hierarchical Report Writer data structures. A node structure exists for each appearance of a box in the final report, but a single module structure exists for a box regardless of the number of times it appears.

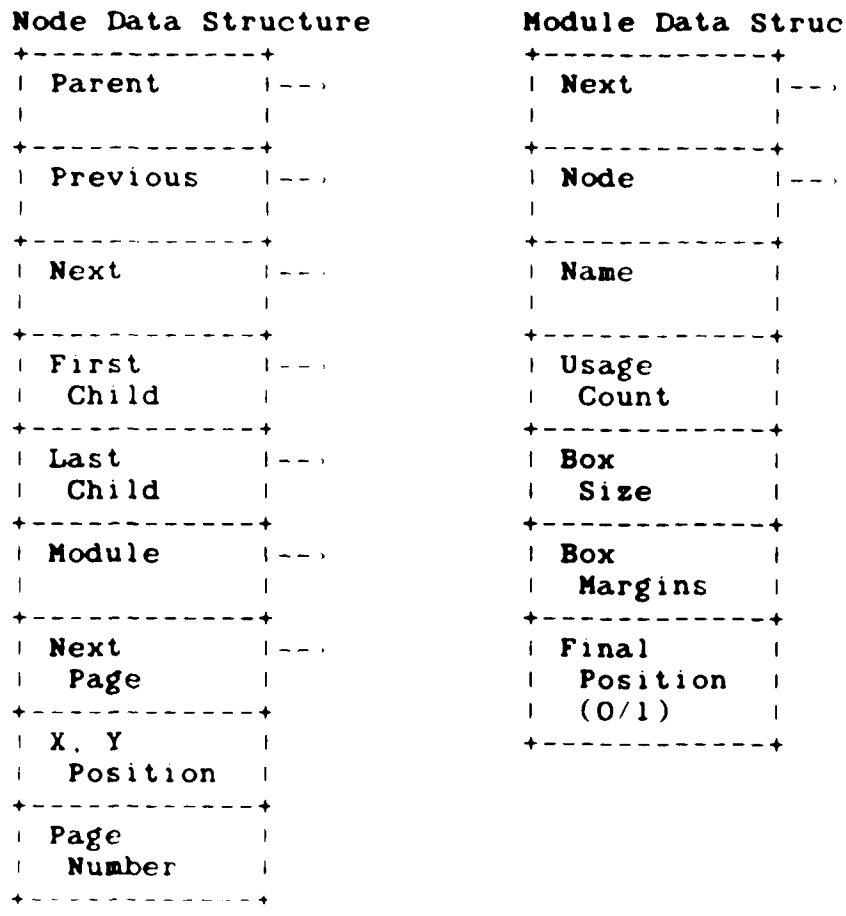


Figure 3-3 Hierarchical Report Writer Data Structures

3.3 Performance Requirements

3.3.1 Programming Methods

A C and COBOL program are used to input the data structures created by compiling the FDL source file with RW and the external schema data structures from the CDM and output the program which produces the report.

3.3.2 Program Organization

The Report Writer is developed in a modular fashion and may be combined with the Rapid Application Generator subsystem of the UI in the future to add functionality.

The Hierarchical Report Writer is implemented as a separate post-processor in order to simplify the interfaces and take advantage of existing code.

3.3.3 Expandability

Since the Report Writer is only one facility of several which comprises the complete Application Generator, it is developed with the other application generator capabilities in mind, and in a sense is a prototype for them.

Post-processors other than the Hierarchical Report Writer may be developed in the future to allow for other types of hierarchical and/or network reports.

3.4 Data Base Requirements

The Report Writer Generator accesses the CDM data dictionary information (the data about the data stored in the CDM) to generate neccessary application code for the External Schemas. It also accesses the Form Processor data structures to generate necessary application code for processing the Presentation Schemas.

3.4.1 Sources and Types of Input

CDM - Provides characteristic information about the External Schema items being used by the application. The machine representation, the size, and number of positions after the decimal point are retrieved from the CDM by the Report Writer Generator.

Form Definition Objects - Provides Characteristic information about the Presentation Schema Items. The machine representation and the picture format are retrieved from the Form Definition objects by the Report Writer Generator.

3 4 2 Destination and Types of Output

Generated External Schema COBOL record structures - The Report Writer Generator uses the information from the CDM to create the COBOL record structures for the External Schemas being used by the application program These structures are part of the generated application code source file.

Generated Presentation Schema COBOL record structures - The Report Writer Generator uses the information from the Form Definition Objects to create the COBOL record structures for the Presentation Schemas being used by the application program These structures are part of the generated application code source file

Generated Machine Representation Conversion code - The Report Writer Generator uses information in the CDM and Form Definition Objects to create the correct code to go from the External Schema to the Presentation Schemas and vice versa This code is part of the generated application code source file

SECTION 4

QUALITY ASSURANCE PROVISIONS

4 1 Introduction and Definition

"Testing" is a systematic process that may be preplanned and explicitly stated. Test techniques and procedures may be defined in advance and a sequence of test steps may be specified. "Debugging" is the process of isolation and correction of the cause of an error.

"Antibugging" is defined as the philosophy of writing programs in such a way as to make bugs less likely to occur and when they do occur, to make them more noticeable to the programmer and the user. In other words, as much error checking as is practical and possible in each routine should be performed.

4 2 Computer Programing Test and Evaluation

The quality assurance provisions for test consists of the normal testing techniques that are accomplished during the construction process. They consist of design and code walk-throughs, unit testing, and integration testing. These tests are performed by the design team. Structured design, design walk-through and the incorporation of "antibugging" facilitate this testing by exposing and addressing problem areas before they become coded "bugs".

The integration testing entails generating a report application, precompiling the program with CDMP, compiling the resultant program, and producing a printed report using a CDM data base.

Each function is tested separately, then the entire sub-system is tested as a unit. All testing except for integration with software belonging to other companies is done at SDRC on the VAX.

SECTION 5

PREPARATION FOR DELIVERY

The implementation site for the constructed software is the Integrated Support System (IISS) Test Bed site located at General Electric, Albany, New York. The software associated with each CPCI release is delivered on a media which is compatible with the IISS Test Bed. The release is clearly identified and includes instructions on procedures to be followed for installation of the release. Integration with the other IISS CPCI's is done on the IISS Test Bed on a scheduled basis.

APPENDIX A

INSTANTIATION RULES

1) The form hierarchy is traversed in a manner similar to a Depth First Search (DFS):

   a) All items which can be reached from this form by NOT going thru windows or open ended arrays are visited.

   b) All open ended arrays which can be reached from this form by NOT going thru windows or opened arrays are visited.

   c) A data record is read.

2) Upon visiting an item a value from a data record may be obtained.

3) Upon visiting an open ended array:

   a) An element is added.

   b) A DFS is performed on the element.

   c) Repeat from a).

4) Step 3) terminates when:

   a) Adding an element would cause an overflow.

   b) The data reachs the end of a group (nested selects) or the end of the data.

   c) Visiting an item would change its value and said item has a change condition on it.

Notes:

1) Run time relative positioning of fields has not been implemented in version 2.0 of the Report Writer. This capability would enable a field to be positioned relative to another field whose size is determined at run time (e.g. A field containing a summary could be positioned below an open ended array which repeated vertically). In previous versions of this DS a field positionally dependent on another would

require the other field to be instantiated prior to the dependent field.

2) Since the SELECT action is essentially a static specification of the query the Report Writer does not check, at run time, that a field used in a WHERE clause has been instantiated.

APPENDIX B

SEMANTICS OF CONDITIONS AND ACTIONS

Action lists associated with conditions are executed in an order determined from the order of the occurrence of the conditions, the order of the actions within each action list, and the precedence of the actions. The following rules define how each of these factors contributes to determining that order.

1) Conditions are set during the PRESENT action only. Referring to Appendix A:

   a) Overflow conditions are checked at step 3a).

   b) Change conditions are checked at step 1c).

2) Actions are exclusive and do not nest.

3) Actions are executed in order of their priority. The priority of actions is determined as follows:

   a) Actions are selected for execution from the action lists in the order in which the associated conditions were set.

   b) Actions are selected for execution from the action lists in the order of their precedence. This rule modifies rule a). The precedence of the actions is:

      SELECT
      SIGNAL
      PRESENT
      SET
      PAGE

   c) Actions within an action list are selected for execution in the order in which they appear in the action list.

APPENDIX C

REPORT WRITER SYNTAX RULES

The following is the syntax for the Report Definition Language.

Report Definition

```
CREATE REPORT report_name

     [ Form_Definition ] ...

     [ Condition_Definition ] ...

     [ Graph_Definition ] ...
```

Form Definition

```
CREATE FORM form_name

     [ CONDITIONAL]

     [ PROMPT Location prompt_string ] ...

     [ Field_Definition ] ...
```

Field Definition - Items

```
ITEM item_name [ Repeat_Spec ]

     Location

     [ SIZE cols [ BY rows ] ]

     [ VALUE {string_constant} ]
             {'._DATE'         }
             {'._TIME'         }
             {'._PAGENO'       }

     [ NODUP ]

     DISPLAY AS { OUTPUT }
                { TEXT   }

     [ DOMAIN ([LEFT ] [UPPER] [ PICTURE picture_spec ])]
              [RIGHT] [LOWER]

     [ SUMMARY [BY item_name-1... ] statistic item_name-2... ]

     [ PROMPT Location prompt_string ...]

     [ LINE FROM location-1 TO location-2 ... ]
```

Field Definition - Forms

```
FORM form_name [ Repeat Spec ]

     Location

     SIZE {cols} [ BY {rows} ]
          { *  }      { *  }

     [ PROMPT Location prompt_string ] ...

     [ LINE FROM location-1 TO location-2 ... ]
```

Field Definition - Windows

```
WINDOW window_name [ Repeat Spec ]
     Location
     SIZE int [BY int ]
     [ PROMPT Location prompt_string ] ...
```

Field Definition - Graphs

```
GRAPH graph_name

+--          --+
|{ABSOLUTE}    |
|{ADDITIVE}    |
+--          --+

+--                                                          --+
|AXIS {HORIZONTAL}                                             |
|     {VERTICAL  }                                             |
|   +--                                                  --+   |
|   |{LABEL  [FONT font] [SIZE size] [COLOR color]}      |   |
|   |   "string"}                                        |   |
|   |{TICK ndiv [minor]                                  |   |
|   |     [FONT font] [SIZE size] [COLOR color]          |   |
|   |      "string"}                                     |   |
|   +--                                                  --+   | ...
|   [MINIMUM lower_limit]                                      |
|   [MAXIMUM upper_limit]                                      |
|   +--              --+                                       |
|   | SCALE {LINEAR} |                                       |
|   |       {LOG10 } |                                       |
|   |       {LOG   } |                                       |
|   +--              --+                                       |
+--                                                          --+

[BACKGROUND color]

+--          --+
|GRID {YES } |
|     {NO  } |
|     {FINE} |
+--          --+

[LEGEND Location [BOX]]

Location

+--                      --+
|MARGIN {LEFT } margin    |
|       {RIGHT} margin    | ...
|       {ABOVE} margin    |
|       {BELOW} margin    |
+--                      --+
```

```
+--                                                              --+
|PROMPT Location [FONT font] [SIZE size] [COLOR color]           |
|       prompt_string                                            |
+--                                                              --+

SIZE cols BY rows

+--          --+
|TYPE {BAR }   |
|     {PIE }   |
|     {PLOT}   |
+--          --+
```

Repeat Spec

```
+--                                                              --+
| ( * { HORIZONTAL } [ WITH int SPACES ] [ , ... ] )             |
|     { VERTICAL   }                                             |
+--                                                              --+
```

Location

```
                /                                                \
               | { [ int ] { LEFT  } OF [ field_name ] }  +-      |
               | {          { RIGHT }                   }  | AND  |
               | { COLUMN int                           }  +-     |
               |                                                  |
               |       { [ int ] { BELOW } [ field_name ] } -+    |
               |       {         { ABOVE }                 }  |   |
               |       { ROW int                           } -+   |
               |                                                  |
               | { [ int ] { ABOVE } [field_name ] }  +-          |
               | {         { BELOW }               }  | AND       |
               | { ROW int                         }  +-          |
[Rpt] AT       <                                                  >
               |       { [ int ] { RIGHT } OF [ field_name ] } -+ |
               |       {         { LEFT  }                  }  | |
               |       { COLUMN int                         } -+ |
               |                                                 |
               | { [ int ] { LEFT  } OF [ Rpt OF ] [ field_name ] } |
               | {         { RIGHT }                              } |
               |                                                 |
               | { [ int ] { ABOVE } [ Rpt OF ] [ field_name ] }  |
               | {         { BELOW }                          }   |
               |                                                 |
               | int-1 int-2 [RELATIVE TO [Rpt OF] [field_name] } |
                \                                                /
```

Rpt

```
     /              \
    | TOP LEFT      |
    | TOP            |
    | TOP RIGHT      |
    | LEFT           |
    < CENTER         >
    | RIGHT          |
    | BOTTOM LEFT    |
    | BOTTOM         |
    | BOTTOM RIGHT   |
     \              /
```

Condition Definition

```
      /                                                \
     | (OVERFLOW BY field_name) Condition_Action ...    |
ON   < (CHANGE item_name) Condition_Action ...          >
     | (STARTUP) Condition_Action ...                   |
      \                                                /
```

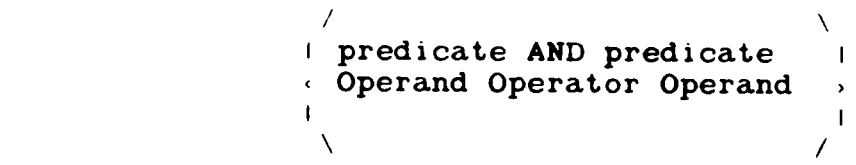
Condition Action

```
      /                                    \
     |  PAGE                                |
     |  PRESENT {field            }         |
     |          {form [IN window]}          |
     <  SET item_name = constant            >
     |  SIGNAL OVERFLOW BY field_name       |
     |  Select_action                       |
      \                                    /
```

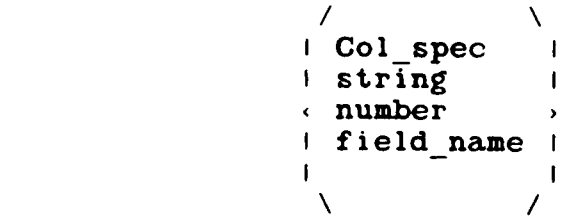
Select action

```
SELECT  qualified_name = Col_spec ...
        [ DISTINCT ]
        [ FROM table [ abbreviation ] ,... ]
        [ WHERE Predicate ]
        [ ORDER BY Col_spec { ASCENDING  } ... ]
                            { DESCENDING }
        [ '{' Select_action ... '}' ]
```

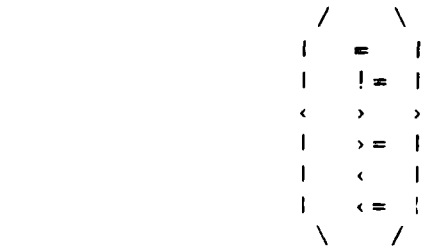
Predicate



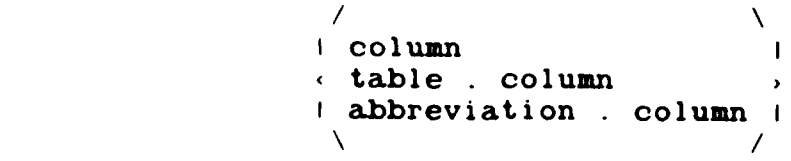
Operand



Operator

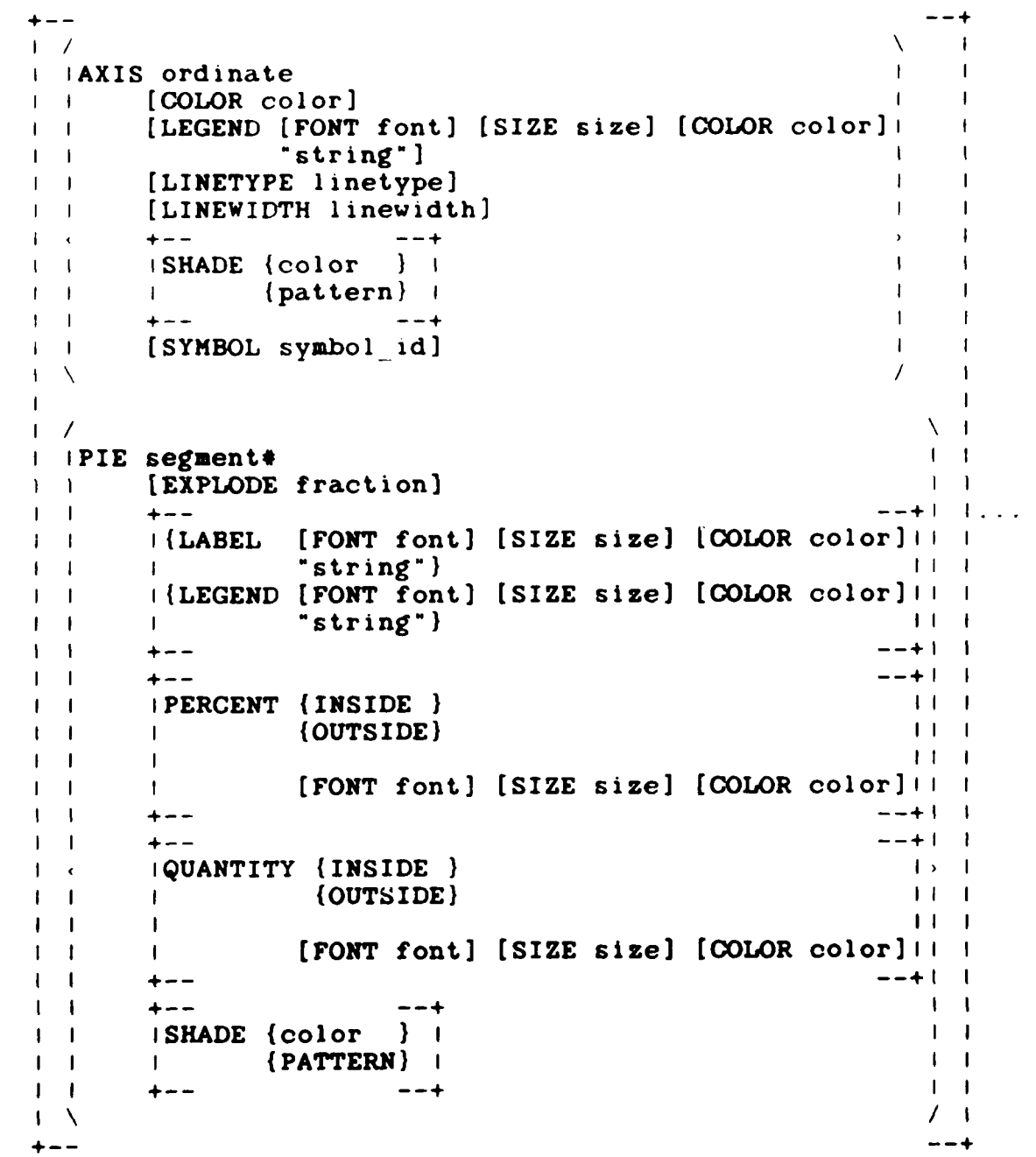


Col_spec



Graph Definition

```
CREATE GRAPH graph_name
     USING ordinate [[VERSUS] abscissa] ...
```



U.S. Government Printing Office 1987 — 748-061/60961

END

8-87

DTIC